by
Jay Munro

Lab Notes

Using HPPCL to Get The Most from Your LaserJet Printer

■ When you need to go beyond the printer driver support built into your applications, QuickBASIC offers one handy way to tell your printer what to do.

"A LaserJet in every office" might once have seemed as empty as a campaign promise, but if you include compatibles, it's a prospect that comes nearer to realization every day. Speed, multiple fonts, graphics capabilities, and high-quality output have made HP's LaserJet a natural choice for almost any printing project. So great is the popularity of this type of printer, that virtually every word processing package and software application that outputs to a printer now supports at least some of its features.

But where do you start when you want to make use of features not normally supported by your software or to add HP LJ support to your own programs? This installation of Lab Notes will help you understand the cryptic LaserJet language and, by means of some simple examples written in QuickBASIC, will show you how to incorporate it into your own programs. LZSELECT.EXE, in this issue's Utilities column, will simplify the task of sending the necessary commands to your LaserJet, even if you have no interest in programming.

Rudimentary printer languages have existed as long as there have been printers for computers to send output to. In the early days of microcomputing, printer commands were limited to *control characters*, borrowed from the teletype world. Standard low-value ASCII codes for commands like carriage return, line-feed, and form-feed were generated by simultaneously pressing the Ctrl key and an alphabet key. Ctrl-M, whose ASCII (decimal) value is 13, caused the printer to execute a carriage return, for example.

Only 32 such control codes were available, however, so *escape codes* were added to the language to support a growing number of printer commands. Implementing control with an escape character was easy: the printer would watch the incoming data stream, and when it found an ESC character (ASCII 27), it would appropriate the next character for itself, look it up in an internal table, and execute the designated command. All other characters in the data stream would pass through untouched and be printed.

As printers became still more sophisticated, escape codes went from simple two-character sequences (ESC plus one other character) to multicharacter escape sequences. By now, the escape sequences had not only to support the basic functions, but to accept additional parameters and data, as well.

ENTER THE HP LASERJET PRINTER

In 1984, Hewlett-Packard Co. introduced the machine that would change the future of printers forever. Admittedly, the original HP LaserJet looked and sounded more like an office copier than a printer. Inside, however, it boasted features far beyond those of any previous consumer printer. And to execute the LaserJet's set of more than 60 commands, HP introduced HPPCL, the HP LaserJet Printer Command Language. HP's PCL has been the standard for all HP LaserJet printers produced since. Indeed, like the Hayes modem command set, HPPCL has become a *de facto* standard that many other printer companies find it necessary to support.

The HPPCL standard began as an expandable language that could be used for both impact and nonimpact printers. It has since evolved into what is currently a five-level structure of printer commands. Each higher level builds on the features implemented in the previous level and provides almost total backward compatibility. (The only exceptions seem to be on undocumented features.) Level I, Print and Space, consisted mostly of the control codes, reset, simple character-set selection, and basic raster graphic support. Level II, Electronic Data and Transaction Processing, added cursor, margin, and line-spacing control. The original HP LaserJet supported what is now called Level III, or Office Word Processing. Level IV, Page Formatting, is supported by the HP LaserJet 500, Plus, Series II, and IIP. At the top of the scale, supported by the HP LaserJet III, is Level V, Office Publishing. This level not only supports more advanced PCL commands, but also incorporates HP-GL/2, the HP Graphics Language used on all HP plotters.

To cover the widest range of printer support, this article will focus primarily on Level IV, so unless otherwise noted, all the examples presented here will work with the HP LaserJet Series II, IIP, IID, III, and IIID printers. A couple of examples will give at least a taste of Level V (and so will only run on the HP LaserJet III or IIID printers), but these will be well labeled.

Two points should be noted here. First, while the PCL level designates the set of

available commands, not all models that use the same general PCL level have identical features and commands. For example, with the HP LaserJet Series II printer, you need separate fonts for portrait (vertical paper orientation) and landscape (horizontal paper orientation) printing. With the HP LaserJet IIP printer, on the other hand, a single downloaded font can be automatically rotated when the orientation is changed, so only one font file per size and typeface is needed for either orientation. Thus, knowing the basic PCL level does not eliminate the need to become familiar with the features of your specific model. Second, if you have not yet purchased an HP LaserJet and are trying to decide between the HP LaserJet IIP or III printer, my recommendation is to go for the model III if you can afford it. The advantages will be obvious later.

PCL COMMAND STRUCTURE

The HP *LaserJet Technical Reference Manual* for your particular printer is your best source for HPPCL information and HP LaserJet printer programming in general. These manuals are not provided with the printers, since many users never go further than what their application software provides. You can obtain them from HP, at a cost of $25 for the Series II and at no cost for the HP LaserJet IIP, III, or IIID, by mailing the card that comes with your printer.

Within the manuals, the escape sequences that constitute the commands of the HP Printer Command Language are listed in groups by function. The major headings may vary in exact wording for different models, but the following list (for the HP LaserJet IIP printer) is typical:

- Job Control Commands
- Page Control Commands
- Cursor Positioning
- Font Selection
- Font Management
- Soft Font Creation
- Graphics
- Macros
- Programming Hints

Each category is further divided into subheadings, and under each heading are the escape sequences that execute the commands. There may be hundreds of escape codes for a particular printer model, but they all share the same structure.

The HPPCL commands, usually referred to as *escape sequences,* can be divided into two types: *two character* sequences and *parameterized* sequences. The two-character escape sequences obviously have a very simple format: an escape character followed by an ASCII character in the range from 0 to ~. (This range corresponds to CHR$(48) through CHR$(126) in BASIC.) Two examples are the ESCE (Printer Reset) and ESC= (Half-Line-Feed) commands.

Most PCL commands are Parameterized escape sequences. The parameterized format uses a group of characters, divided

Not all models that use the same PCL level have identical features and commands.

(without intervening spaces) into five sections. For example, the sequence

```
Esc&a#L
```

where ESC is the escape character and a suitable number replaces the #, is used to set the left margin.

Figure 1 shows how the five main areas of a parameterized escape sequence are denominated in the HP documentation. The meanings are as follows:

Escape character—The first character in the command sequence is always the Escape character, which signals the printer that one or more commands are to follow. The representation of the escape character however, can be confusing—even within HP's own documentation. In the HP documentation the escape character is shown as ᵉC, as in ᵉC&110. However, on a LaserJet III font printout, the escape is shown as [esc]. We will be referring to the escape character here either as ESC, when talking about HPPCL escape sequences, or as ESC$, when showing BASIC program code. In any case, when ᵉC, ESC, or ESC$ is shown, it is representing the ASCII character 27, which is actually sent to the printer. In BASIC this character is represented by a CHR$ (27). When an escape character is printed to the screen, it shows up as a left arrow, (←).

Parameterized character—The Parameterized character is selected from the range of ASCII characters from ! to / (CHR$(33) to CHR$(47) in BASIC or !"#$%&'()*+,-./). Its purpose is to tell the printer that this escape sequence is a parameterized command; that is, that it has other characters associated with it. This character differentiates the code from a two-character escape sequence.

Group character—The Group character defines the group or type of comman to be performed. In conjunction with the parameterized character, the group character specifies the type of function to be performed. The group character is taken from the ASCII character range from ' to ~(that is, CHR$(96) to CHR$(126)). Note that in current HP documentation, the lowercase letter *l* is printed in a script font to avoid confusion with the numeral *1* character. If a letter is used as the group character, it must always be lowercase.

Value field—The Value field specifies a numeric value used by the command.

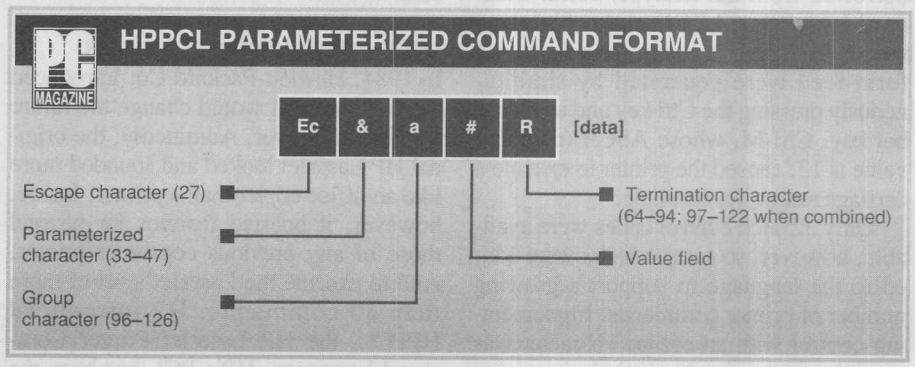

Figure 1: A *parameterized* command in HPPCL consists of these five main components, represented by the corresponding ASCII values. Values 48 to 63 cannot be used, since they are numbers and cannot be capitalized. Note that no spaces may be inserted between the various parts of the command.

This number may be selected from a standard set of choices or it may be a variable, such as a margin setting. Depending on the command, the value field may also accept a + or − and, in some cases, it may be a floating-point number. When a value is not supplied in an escape sequence that normally expects one, the value is assumed to be zero.

Termination character—The Termination character comes from the ASCII range of @ through ^, which is CHR$(64) to CHR$(94) in BASIC. The termination character both specifies the parameter to which the previous value field applies and signals the end of the escape sequence. Note that when escape sequences are combined (as explained later), the termination character is sometimes uppercase and sometimes lowercase.

[*Data*]—In addition to the five areas of the parameterized escape sequence, some commands require sending binary data to the printer. In such cases the value field indicates the number of bytes of data that ll follow the termination character. The raster graphics command is one sequence that precedes a string of binary data.

USING ESCAPE SEQUENCES

Whether of the two-character or of the parameterized type, there are several ways to send the PCL-command escape sequences to the printer. With the exception of graphics and font files, the HP LaserJet printer accepts commands and data in the form of ASCII strings. Many of today's word processors allow user-entered printer escape sequences to be sent from within a document as "printer instructions." Alternatively, ASCII editors and word processors that can produce the ESC character can be used to create a file consisting of nothing but the command sequence. You could, for example, create a file called LM10, whose whole content would be

```
Esc&a10L
```

This file can then be sent to the printer with the DOS COPY command. Thus, to set the left margin on the LaserJet to column 10, you would simply enter

```
COPY LM10 LPT1:
```

This method of LaserJet control was discussed in a four-part Lab Notes series in our February 16, February 29, March 15, and March 29, 1988, issues.

In the examples that follow, we will use still another method, namely, sending the codes to the printer via QuickBASIC programs. QuickBASIC is an easy language to use in experimenting with escape sequences, since what you need is provided—string handling, variable conversion, input, output, and so on. GW-BASIC or BASICA could also be used, though some of the specific BASIC commands employed here would have to be modified, and line numbers would have to be added.

Other than the CHR$(27), the data we send here will be in a quoted string format. For the sake of clarity, and to reduce the number of characters on a line, the string variable ESC$ will be assigned to the escape character, CHR$(27), in all examples. When your own programs require numeric values, you can convert any numbers into strings using BASIC's STR$() function. Note that it is good programming practice to use LTRIM$() to trim the leading spaces that QuickBASIC's STR$ function leaves on positive numbers before using the value in an escape sequence.

A short QuickBASIC program for setting the left margin might begin

```
ESC$ = CHR$(27)
INPUT "Enter left margin value
";X% 'prompt for input
```

```
LM$ = LTRIM$(STR$(X%))
 'convert to a string
```

```
LPRINT ESC$;"&a";LM$;"L";
 'build escape sequence
```

In the above program you could eliminate the input and trimming functions and substitute an escape sequence the printer can understand directly. In such a case, to send a left margin of 10 columns, the second and final line of the program would simply need to be

```
LPRINT ESC$;"&a10L";
```

Notice that when printing combined strings, we use semicolons to add the next character. The same sequence could have been sent as

```
LPRINT ESC$+"&a10L";
```

FIGURE_2.BAS — **COMPLETE LISTING**

```
WIDTH LPRINT 255                          'disable QB's width checking
ESC$ = CHR$(27)
INPUT "Enter a Left Margin Value  (0 to 80) ";LM%
INPUT "Enter a Right Margin Value (0 to 80) ";RM%
LM$ = LTRIM$(STR$(LM%))                   'convert value to string
RM$ = LTRIM$(STR$(RM%))                   'convert value to string
Temp$ = ESC$+"&a"+LM$+"l"+RM$+"M" 'create escape sequence
LPRINT ESC$;"E";                          'reset printer
LPRINT Temp$;                                        'print new margins
FOR X = 1 TO 5
    LPRINT STRING$(160,65);               'print sample characters
NEXT X
LPRINT CHR$(12);                          'print formfeed
```

Figure 2: This short program illustrates how two escape sequences, for setting the left and the right margin, can be combined in HPPCL and safely concatenated in QuickBASIC and sent to the printer.

This method is not generally recommended, however, because QuickBASIC must make a temporary string of the codes and then use LPRINT to send it to the printer. This takes more time. Further, QuickBASIC occasionally has difficulty printing complex concatenated strings. If your program requires concatenating strings, create a temporary string first, and then LPRINT that. Figure 2 shows an example of this method and deserves some detailed examination.

When using the LPRINT command, QuickBASIC sets the width of printer output to a default of 80 characters. The statement WIDTH LPRINT 255 is used to disable the default, preventing possible breakup of a long escape sequence. The LPRINT statement also automatically inserts a CR-LF combination unless a semicolon is placed at the end of the line. When sending escape sequences, then, always be sure to add a semicolon to the end of the LPRINT statement. This way you can avoid unwanted line-feeds.

The two input statements in Figure 2 prompt for integer values of LM and RM, for left and right margin, respectively. The integer values are then converted to strings and the leading spaces trimmed. A tempo-

rary string, Temp$ is created by concatenating or adding the strings together with the escape character, ESC$, at the beginning. Temp$ is then printed with the LPRINT command, along with a string of sample characters and a form-feed.

Turning from considerations about using QuickBASIC to those that concern HPPCL, you should notice that the Temp$ in Figure 2 contains *two* PCL commands—one for the left and the other for the right margin setting. There is only one ESC$ and only one pair of parameterized and group characters, however. This is because PCL allows *certain* escape sequences to be combined, thus saving printer memory and reducing the transfer time required.

The PCL rules for combining escape sequences are simple: When the first two characters following the escape character are the same, the codes can be combined by dropping the escape, parameterized, and group characters from all but the first code in the sequence; and second, the termination character of all but the *last* code must be lowercase—the last termination character must be uppercase. In QB, the *separate* strings for setting a left margin of 5 and a right margin of 75 would be

```
ESC$+"&a5L"
```

and

```
ESC$+"&a75M"
```

When combined, however, they become

```
ESC$+"&a5175M"
```

(Outside of BASIC, in PCL terms, the combined sequence is ESC&a5175M.)

A common mistake when combining escape sequences is trying to combine unlike codes. The left margin command ESC$;"&a5L"; could not be combined with the top margin code ESC$;"&l6E"; because the first two characters—&a and &l—are not the same case. Again, a problem is created by not using correct case for group and termination characters, as in

```
ESC$;"&A5L75M"
```

You'll also encounter difficulties if you include the parameterized and group character more than once in a combined sequence, as in

```
ESC$;"&a5L&a75M"
```

To fix this sequence, you could add an ESC$; after the L and before the &. But it is more economical to make the L lowercase and drop the second set of &a characters.

The usual symptom of these problems is that the LaserJet ignores the commands and simply prints the escape codes instead of interpreting them.

If you find that the printed output does not match what you expected or the escape sequences were ignored, send a Reset command, as shown in the next program, to the printer, before attempting to resend.

INITIALIZING THE HP LASERJET

The miniprogram shown in Figure 3 initializes the printer for a common task, namely, to print a series of reports. The program starts by sending a Reset command:

```
ESC$;"E";
```

The Reset escape sequence prints anything that is in the printer buffer, clears out any temporary soft fonts and temporary macros in the printer's memory, and restores the printer to the *user's* default environment. The Reset command does not return to the factory defaults, but to the known conditions the user has set from the front panel. Since an escape sequence always takes precedence over a front-panel setting, Hewlett-Packard recommends that the Reset command be sent at the beginning of every job.

To save space in the program of Figure 3, the escape sequences have been combined, following the rules just outlined. The left and right margins have been combined as before, and the top margin setting has been combined with a command to disable perforation skipping.

Skipping perforations is not something one would ordinarily associate with sheet-fed printers. Nonetheless, mismanagin the area between the bottom of the text area on one page and the top of the printable area on the next page can produce undesired effects. The default setting is Perforation Skip Enabled. This causes the printer to eject a page when the printing goes beyond the number of lines set for the page length. While this is normally what you want the printer to do, it can be annoying when you print documents that have embedded page feeds and that are formatted to contain exactly the number of lines to which the page length is set. The result in such a case is that an extra page is ejected for each one printed.

By sending the sequence

```
ESC$;"&l0L";
```

the program in Figure 3 overrides the HP default setting. Be aware, however, that with Perforation Skip Disabled, any text printed past the actual printable area will be lost or truncated, so this setting is desirable only in the case just mentioned. To restore the HP default, just change the 0 in the sequence back to 1; that is, send the command sequence

```
ESC$;"&l1L";
```

A note should be added here about the left and right margin settings, as well. Horizontal Margin settings are relative to the Horizontal Motion Index, or HMI. The HMI can be set directly with the QB escape sequence

```
ESC$;"&k#H";
```

(ESC&k#H in PCL), where # is the number of 1/120-inch increments to be used for each column unit. This value can range from 0 to 840 on the Series II, and up to 32,767 for the HP LaserJet IIP, III, and IIID; it is valid to four decimal places.

When a fixed-pitch font is being used, the default HMI value is set by that font.

INITIAL.BAS — **COMPLETE LISTING**

```
WIDTH LPRINT 255
ESC$ = CHR$(27)
LPRINT ESC$;"E";          'Reset to defaults
LPRINT ESC$;"&l6e0L";     'Top margin = 6, disable perf skip
LPRINT ESC$;"&a5175M";    'Left margin = 5, Right = 75
```

Figure 3: This short program initializes the LaserJet to values suitable for printing reports.

With the normal 10-pitch courier font, a left margin of 5 would be 5 characters, which, at 10 characters per inch, results in a half inch margin. If you change the font to 16.66 pitch—the condensed, or line-printer, font—the margin of 5 characters becomes a little over a third of an inch. Note that the right margin is measured from the left edge of the paper, so to create a 1-inch right margin, you would use a value of 75.

As explained later on, the margins are based on an area known as the Logical Page. So to get a half-inch margin on a LaserJet III, the adjusted margin setting would be approximately 2, not 5 (the Logical Page starts .25 inch from the edge). The Logical Page edge varies in width between HP printer models, so the most accurate setting is achieved by trial and error.

ROTATE.BAS — **COMPLETE LISTING**

```
DEFINT A-Z                  'integers only please
WIDTH LPRINT 255            'disable BASIC's line wrap
ESC$ = CHR$(27)             'assign ESC$
LPRINT ESC$; "E";           'reset printer
LPRINT ESC$; "&a27r45C"; 'position cursor row 27, col 45
FOR X = 270 TO 0 STEP -90 'set-up a for next loop with
   LPRINT ESC$; "&f0S";   'push (save) cursor position
   Temp$ = "&a" + LTRIM$(STR$(X)) + "P" 'build the escape sequence
   LPRINT ESC$; Temp$;    'print escape sequence
   LPRINT STR$(X); " degrees"; 'print some text
   LPRINT ESC$; "&f1S";   'pop (retrieve) cursor position
NEXT X
LPRINT CHR$(12);            'form feed
```

Figure 4: With the LaserJet III and subsequent models, print direction can be rotated in 90-degree increments, without changing page orientation.

ORIENTATION AND PRINT DIRECTION

The default print orientation is Portrait mode, which prints on standard-size paper with the text across the short way. It's called *portrait* mode because a painting of a person usually has a vertical orientation. Conversely, *landscape* mode prints the long way across the paper. It also takes its name from the art world, since landscapes are normally painted horizontally.

Print orientation can be changed either at the front panel or with an escape sequence, but it can be set only once per page. Changing the orientation in midpage causes the printer to eject the current page before honoring the new orientation. The QB escape sequence

```
ESC$;"&l0O";
```

sets portrait mode. Changing the 0 to a 1, that is

```
ESC$;"&l1O";
```

specifies landscape mode. (Again, outside its BASIC representation, the respective PCL escape sequences for portrait and landscape orientations are ESC&l0O and ESC&l1O, respectively.)

The ability to change the printing direction is a feature recently introduced with the HP LaserJet III printer. While setting the paper orientation is still a once-per-page command, the print direction may be changed as often as needed. If you have a LaserJet III or later, the program shown in Figure 4 will demonstrate the print direction command

```
ESC$+"&a#P"
```

where the # is replaced by the degree of rotation, in 90-degree increments.

As the text is rotated, the cursor position stays in the center because of the PUSH and POP cursor-position commands. The Push Cursor ESC$;"f0S"; and the Pop Cursor ESC$;"f1S"; commands are used in pairs to preserve the current cursor position and can be nested up to 20 deep. Try removing them and see what happens: the cursor position is left at the end of each word and, instead of making a four-point star, a box is created.

The ability to change the printing direction is a feature recently introduced with the HP LaserJet III printer.

CURSOR POSITIONING

The program in Figure 4 positions the cursor by row and column, which is similar to printing to a text screen on the PC. The Row measurement is based on current Vertical Motion Index (VMI), which can be set with the escape sequence

```
ESC$;"&l#C";
```

where # is the number of 1/48-inch increments between printed rows. As with the HMI, this can be varied, ranging from 0 to 336 for the Series II, or to 32,767 for the LaserJet IIP, III, and IIID. The VMI can also be set by the lines-per-inch command

```
ESC$;"&l#D";
```

If the lines per inch is at the default of 6 lines per inch, then row 24 is almost 4 inches down from the top. However, if the lines per inch is set to 12, when the cursor is located at row 24 it's only about 2 inches down from the top. Similarly, the column measurement is based on the Horizontal Motion Index (HMI), which is usually the width between fixed-pitch characters. When using proportional fonts, the relationship between columns and the character width will no longer be valid, because of the variable amount of space used by the characters.

In addition to positioning in terms of rows and columns, then, you can also position the cursor by units measured in *dots* or *decipoints*. The *dot* is the smallest printable unit—1/300-inch—on the HP LaserJet. The printer's 300-dot-per-inch (DPI) resolution is the same in both vertical and horizontal directions. While the resolution of the HP LaserJet has been 300 dpi since the first model, the ability to locate the cursor in terms of dots did not become available until the Series II.

Decipoints are the smallest increment of cursor movement. There are 72 points per inch and 10 decipoints per point, so a decipoint is 1/720 of an inch. The LaserJet also supports the decipoint unit for compatibility with high-resolution typesetting devices. For internal calculations, the HP LaserJet Series II uses 1/3600 inch, and the IID, IIP, III, and IIID use 1/7200 inch. How-

ever the decipoint ($^1/_{720}$ inch) is the smallest accessible unit of movement.

The QB program shown in Figure 5 locates and prints a message using cursor positioning specified in dots. When text is located directly with the various cursor-positioning commands, left and right margin settings and text length settings are overridden. This feature can be useful when you need to position footnotes or page numbers and don't wish to reformat the body of text.

In Figure 5 the cursor is first located at row 10. The letter F is then printed repeatedly on the same line, starting at 300 dots, and locating it 20 dots to the right on each pass through the loop. Since the normal spacing between the letters would be 30 dots at 10 pitch (10 cpi x 30 dots = 300 dpi), this produces a partial overlapping of the printed Fs.

The positioning commands discussed so far have located the cursor at an absolute place on the page. The reference location (0,0) for absolute positioning is the intersection, or corner, where the top margin meets the left edge of the logical page. The cursor can also be moved relative to its current position, however. Using relative movement provides an easy way to move in specific increments without having to calculate every point.

To locate the cursor relative to the last position, you add a plus sign to the value for positive movement; likewise, a minus sign for negative movement. To move the cursor 10 dots to the right of the current position, the QB escape sequence would be

```
ESC$;"*p+10X";
```

To move 10 dots to the left, you would simply change the + to a − sign in the sequence shown. Figure 6 lists a short QB program that prints at incremented values for both the Row and Column.

RELCURSR.BAS — **COMPLETE LISTING**

```
DEFINT A-Z
WIDTH LPRINT 255
ESC$ = CHR$(27)
LPRINT ESC$; "E";
FOR X = 0 TO 9              'do 10 Heys!"
  LPRINT ESC$; "&a+"; LTRIM$(STR$(X)); "R"; 'vertical position
  LPRINT ESC$; "&a+"; LTRIM$(STR$(X)); "C"; 'horizontal position
  LPRINT "Hey";             'print a message
NEXT
LPRINT CHR$(12);            'print page
```

Figure 6: By putting a plus sign (+) or a minus sign (−) in the value field, the printing position or cursor can be shifted relative to its present location in terms of both rows and columns.

PRINTABLE AREAS

Any explanation of absolute and relative cursor locations would be incomplete without a discussion of printable areas. The HP LaserJet page contains three areas: the *Physical Page,* the *Logical Page,* and the *Printable Area*. These are shown diagramatically in Figure 7.

The *Physical Page* is the actual paper size—usually the standard letter size paper of 8.5 by 11 inches, though legal, European, and executive sizes are also supported. In no case, however, is it possible to print to every point on the Physical Page with the HP LaserJet.

The *Logical Page* is the area within the Physical Page where the cursor can be located or addressed. The cursor may be positioned anywhere within the Logical Page boundaries, but cannot move outside them. The Logical Page on a LaserJet III is approximately 75 dots (.25 inches) in from the Physical Page edge on the left and right sides. The Logical Page extends to the top-to-bottom edges of the Physical Page in either portrait or landscape orientation.

Margins are measured in relation to the Logical Page not the Physical Page. To set an accurate physical margin, the desired margin must account for the 75-dot difference between the paper's edge and the Logical Page. To set an actual 1-inch left margin, you must set the margin to .75 inch. (1.00 inch − .25 inch = .75 inch). Even though the whole length of the Logical Page can be addressed, you cannot print on all of this area. The top margin default is 3 lines at 6 lines per inch, or .5-inch from the top of the paper, for example.

The *Printable Area* designates the maximum printable boundaries. Characters that extend outside the Printable Area of the Series II are clipped at the characte level (that is, the whole character is lost); where as characters that extend outside the boundaries of the LaserJet IIP, IID, III, and IIID are clipped at the dot level; this means that only a portion of the character is missing. In these latter models, the printable area is normally slightly wider than the Logical Page. This allows the LaserJet IIP and III to use the Left and Top Offset Registration—ESC$;"&l#U"; for the left, and ESC$;"&l#Z"; for the top. The #s are replaced by the number of decipoints by which to offset and are either positive or negative values, to allow a shift in any direction.

The offset registration adjusts the text position of the Logical Page to provide room for a binding, such as three-ring punching or spiral. Note, however, that if the Offset is adjusted too far and so goes outside the Printable Area, characters or data will be lost.

POSITION.BAS — **COMPLETE LISTING**

```
DEFINT A-Z
WIDTH LPRINT 255
ESC$ = CHR$(27)
LPRINT ESC$; "E";          'reset printer
LPRINT ESC$; "&a10R";      'locate cursor at row 10
FOR X = 300 TO 2000 STEP 20   'locate at 300 dots + X
    Temp$ = ESC$ + "*p" + LTRIM$(STR$(X)) + "X"
    LPRINT Temp$; "F";     'print esc code + "F"
NEXT X
LPRINT "ast!";         'print rest of the word
LPRINT CHR$(12);       'print page feed
```

Figure 5: The cursor (print) position on the Series II, IIP, and III printers can be set in increments of 1/300 of an inch, the width of a single dot.

LINES AND FILLS

Drawing lines with the HP LaserJet requires a slight adjustment in the way you think of a line. BASIC programmers normally regard a line as a fixed-width entity When using an HP LaserJet to draw a horizontal or vertical line, you must instead think in terms of a filled rectangle, adjust-

ing its horizontal and vertical width to create it. The program shown in Figure 8 illustrates the procedure.

The printed result of running this program is a 4- by 4-inch cross, each line of which is 10 dots wide. The cross consists

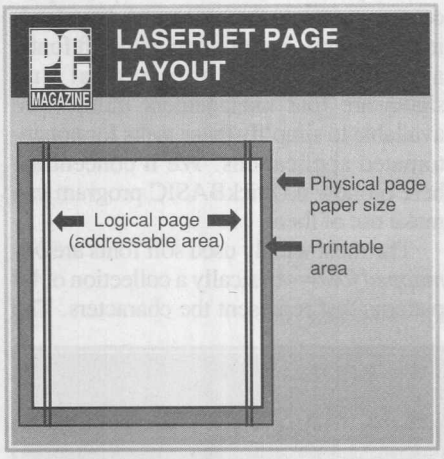

Figure 7: The LaserJet cannot print to the physical edges of an 8.5- by 11-inch sheet. In Series II printers, the width of the "logical" (cursor addressable) and the "printable" areas is the same and totals approximately half an inch less than the paper width (fixed at 50 dots from the left side and 100 dots from the right side). For Series IIP and later models (as shown here) the logical width is centered at 75 dots from each side but can be shifted to either edge of the printable area by using offset commands. In all cases, the logical page length is longer than the printable page length by less than half an inch.

of two rectangles; one is 1,200 dots wide by 10 dots tall, and the other is 10 dots wide by 1,200 dots tall. For the first rectangle, the `ESC$;"*c0P";` sequence selects a solid black fill pattern for the rectangles. For the second, the solid fill escape sequence is combined with the size commands.

Solid black is not the only fill pattern the HP LaserJet can produce. There are seven shades of gray, plus black, and six fill patterns that can also be used. To see a sample of each, simply run the program listed in Figure 9.

The first row of boxes shows the gray scale, whose percentage (from 1 percent to 100 percent) is divided into eight ranges. The second row are the six HP defined patterns (1 to 6). The six patterns are; horizontal, vertical, ascending diagonal, descend-

CROSS.BAS — **COMPLETE LISTING**

```
DEFINT A-Z
WIDTH LPRINT 255          'disable BASIC linefeeds
ESC$ = CHR$(27)           'assign ESC char
LPRINT ESC$; "E";         'reset printer
LPRINT ESC$; "*p1200y600X";      'locate 4" down, 2" over
LPRINT ESC$; "*c1200a10b";       '1200 x 10,
LPRINT ESC$; "*c0P";             'black fill
LPRINT ESC$; "*p600y1200X";      'locate 2" down, 4" over
LPRINT ESC$; "*c10a1200b0P";     '10 x 1200, black fill
LPRINT CHR$(12);          'form feed
```

Figure 8: Lines are drawn in PCL by filling rectangles appropriately. This short program creates a cross, made up of two such rectangles.

ing diagonal, horizontal and vertical cross hatch, and diagonal cross hatch lines. The escape sequence for Area Fill is

```
ESC$;"*c#G";
```

where # is replaced by either a percentage for a fill (gray scale), or a number from 1 to 6 for a pattern. The additional sequence

```
ESC$;"*c#P";
```

then specifies what type of fill to use. The available options are:
0—solid black fill
1—solid white fill
2—shaded (percent on last code)
3—HP defined pattern (1 to 6)
5—current pattern
Fills 0, 2, and 3 are available for all printers; 1 is available on the IIP, III, and IIID and 5 is only available with the III and IIID.

The solid white fill, which is restricted to the IIP and later printers, is used to erase areas. To create a frame with a wide border, you can define a black or shaded box with a white box inside it. The demonstration program listed in Figure 10 will work on a IIP or III printer.

The rectangle fill commands can create only horizontal or vertical lines. To create a diagonal line, a circle, or an arc, the requisite points would have to be calculated and connected with small boxes or lines. Another method would be to create a bit image of the line in memory and send it to the printer in a raster graphics format. However, in PCL Level V, the HP LaserJet III supports the HPGL/2 extension to the PCL command set that contains many predefined graphics commands. HPGL is the Hewlett-Packard Graphics Language, a widely supported plotter language.

HPGL/2 support in the HP LaserJet Series III is a topic for another time, but the program shown in Figure 11 will give you a sample of the power of HPGL/2.

PATTERNS.BAS — **COMPLETE LISTING**

```
DEFINT A-Z
WIDTH LPRINT 255          'disable BASIC linefeeds
ESC$ = CHR$(27)                    'assign ESC char
LPRINT ESC$; "E";         'reset printer
DATA "1","5","15","25","45","65","85","100"  'fill patterns
LPRINT ESC$; "*p300Y";  'locate down 1 inch
FOR X = 1 TO 8
    READ SF$              'read shaded fill
    LPRINT ESC$; "*c300a300b"; SF$; "g2P"; 'print 1" box
    LPRINT ESC$; "*p+300X"; 'move over 1" relative to last position
NEXT X
LPRINT ESC$; "*p+600y270X"; 'move down
FOR X = 1 TO 6
    LPRINT ESC$; "*c300a300b"; LTRIM$(STR$(X)); "g3P";
    LPRINT ESC$; "*p+310X"; 'move cursor
NEXT X
LPRINT CHR$(12);          'form feed
```

Figure 9: LaserJet series printers can produce seven gray shades in addition to black and six fill patterns. Run this program to see an example of each.

SOFT FONTS

LaserJet users who want their documen to have just the right *look* have an enormous variety of downloadable *soft fonts* from which to choose. In addition to purchasing from commercial soft font vendors (HP, Bitstream, Agfa-Compugraphic, and many others), you can even download shareware font samples from the libraries of the HP forum on CompuServe.

Many of today's advanced word processors automate the process of downloading, selecting, and printing with soft fonts, and a number of commercial and shareware font management utilities are available to simplify these tasks for nonautomated applications. We'll concentrate here on how a QuickBASIC program can make use of them.

The most widely used soft fonts are *bitmapped fonts*—basically a collection of bit patterns that represent the characters. The bit patterns are organized by internal escape commands that define the character, size, and other attributes of the character set. The font is stored in the printer's memory by downloading or transferring the data from a file. Each soft font is identified within the printer's memory by assigning it a unique ID number, from 0 to 32,767, at download time.

Scalable fonts use hinting to ensure the highest-quality font appearance no matter what size is used.

In the program shown in Figure 12, a soft font is downloaded with an ID of 54, an arbitrary number. For the font filename (FontFile$) you can substitute the name of any HP-compatible soft font you have. The first sequence—ESC$;"*c54D";—sets the ID number to 54. The font file is then OPENed for BINARY as #1, and the printer is OPENed for BINARY as #2. Binary file access is used because the font files contain data with the full range of ASCII values, from 0 to 255. Binary rat than standard character output to the printer is necessary: otherwise DOS would stop downloading if it hit a bit pattern corre-

Lab Notes

ERASE.BAS — **COMPLETE LISTING**

```
DEFINT A-Z
WIDTH LPRINT 255
ESC$ = CHR$(27)
LPRINT ESC$; "E";
LPRINT ESC$; "*p900y600X";        'locate 3" down, 2" in
LPRINT ESC$; "*c900a900b70g2P"; '70% gray box
LPRINT ESC$; "*p+75y+75X";        'locate .25" over and down
LPRINT ESC$; "*c750a750b1P";      'white box
LPRINT CHR$(12);
```

Figure 10: The LaserJet IIP and later printers can "erase" an area by filling it with a white box.

sponding to CHR$(26), the End Of File marker.

The program reads a block of data from the font file into the BUFFER$ with the GET command, and then PUTs the buffer into the printer, repeating until finished. Upon completion of the file transfer, the escape sequence—`ESC$;"c5F";`—is sent to designate the font as permanent. To make the font temporary, you would substitute a 4 for the 5 in this code. A printer Reset command will delete a temporary font but not a permanent one. The newly downloaded font is specified as the primary, or current, font by sending the `ESC$;"(54X";` sequence.

To change between several soft fonts that have been downloaded, substitute the other ID numbers for the 54 in this example. If you download a font using the ID of a font already in printer memory, the new font will replace the old one.

Starting with the LaserJet III, Hewlett-Packard integrated Agfa-Compugraphic's Intellifont font-scaling technology. The LaserJet III contains two resident *scalable* font typefaces, the sans serif Univers, and the serifed CG Times. Additional scalable fonts are available, both in cartridges and in downloadable form.

Scalable fonts have a number of important advantages over the standard bit-mapped fonts. First, scalable fonts can be sized in .25-point increments from .25 to 999.75 points, without using any additional memory. The maximum size of a bit-mapped downloadable font is 200 points (a height of about 3 inches)—memory permitting; the amount of memory your printer has determines the largest size of font that can be downloaded. Second, separate bitmapped fonts must be downloaded for each size and typeface, and this can quickly require a large investment in printer memory.

Moreover, since the scalable fonts also take advantage of the LaserJet III's resolution-enhancement technology, they appear sharper than most outline-generated fonts. When it uses resolution enhancement, the printer actually varies the size of dots to fill in curved and diagonal lines, where the "jaggies" usually appear. The scalable fonts also use a software technique called *hinting,* which ensures the highest-quality font appearance no matter what size is specified. The commands to access hinting are not publicly available, but developers may contact Agfa-Compugraphic to license the technology.

When the Series III does a test printout, the escape sequences needed to access the fonts are printed below the print sample on the right side of the page. Here is an example of how the Univers Bold font sequence appears:

```
<Esc>(10U<Esc>(s1p__v0s3b4148T
```

If each command that makes up the sequence were sent individually, it would break down like this:

```
Esc(10U     -- Select PC 10 character
               set
Esc(s1P     -- Designate proportional
               spacing
Esc(s#V     -- Height in points (#)
Esc(s0S     -- Upright style
Esc(s3B     -- Bold stroke weight
Esc(4148T   -- Univers font type face
```

The underscore would be filled in with

PLOTTER.BAS — **COMPLETE LISTING**

```
'Example of HPGL/2 special effects
WIDTH LPRINT 255: ESC$ = CHR$(27)
LPRINT ESC$; "%1B";                  'enter HP-GL/2 mode
LPRINT "IN;SP1;";                    'initialize
LPRINT "PA2000,4700;";               'select pen 1
LPRINT "PD;PR 0,2550,4200,0,0,-2550,-4200,0;"; 'make a box
LPRINT "PU;";                        'lift pen for move
'define a user fill pattern--index 1, 32 dots wide * 24 dots high
LPRINT "RF1,32,24,";
LPRINT "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,1,0,0,1,1,1,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,";
LPRINT "0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,";
LPRINT "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,";
LPRINT "0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;";
LPRINT "FT11,1;";                    'select above pattern for fill type
LPRINT "SD1,21,2,1,4,200,5,0,6,3,7,4148;SS;"; 'select Univers 200pt
LPRINT "PA2500,5000;";             'move pen to center,
LPRINT "DT*;"                      ' define label terminator as *
LPRINT "PD;";                        'pen down
LPRINT "CF3,1;LBPC*"                 'select user pattern, print PC
LPRINT ESC$; "%0A";                  'switch back to PCL mode
LPRINT CHR$(12);                     'form feed
```

Figure 11: By supporting the HP-GL/2 extension, PCL Level V (restricted to the LaserJet III and later) lets you emulate the output of a plotter.

DOWNLOAD.BAS — **COMPLETE LISTING**

```
DEFINT A-Z                  'font copy routine
ESC$ = CHR$(27)                 'assign Esc character
BufLen = 4096                   'set a buffer size
Buffer$ = SPACE$(BufLen)        'buffer for transfer
FontFile$ = "BW140BPX.USP"      'font file name
OPEN FontFile$ FOR BINARY AS #1 'open the font file
FILELEN& = LOF(1)               'find out it's length
OPEN "LPT1" FOR BINARY AS #2    'open printer
FontID$ = ESC$ + "*c54D"        'set font id number
PUT 2, , FontID$                'send it to the printer
DO                              'download the font
  IF FILELEN& < BufLen THEN     'for the last block
     Buffer$ = SPACE$(FILELEN&) '  it's almost always
  END IF                        '  shorter than BufLen
  GET 1, , Buffer$              'get a block
  PUT 2, , Buffer$              'put it to the printer
  FILELEN& = FILELEN& - LEN(Buffer$) 'track size
LOOP WHILE FILELEN&             'keep going until out of data
Perm$ = ESC$ + "*c5F"           'make it permanent
PUT 2, , Perm$                  '
SetID$ = ESC$ + "(54X"          'make it primary
PUT 2, , SetID$
CLOSE                           'close the files
FOR X = 1 TO 10                 'test it...
LPRINT "This is a test "
NEXT X
LPRINT CHR$(12);                'form feed
```

Figure 12: This program will download any HP-compatible soft font whose filename you supply as FontFile$ (line 4), give permanent status, and assign the (arbitrary) ID number 54.

the point size you want to use. To send the sequence in QuickBASIC for a 14-point font, you simply use the command

```
LPRINT ESC$;"(10U";ESC$;"
(s1p14v0s3b4148T";
```

MACROS

Macros serve the same function in HP PCL that they do in software applications: they speed and simplify operations by eliminating the need to rekey long, frequently needed sequences. As a conclusion to this Lab Notes, we will draw from our previous examples to create a macro program that sets up and prints a letterhead on every page.

The HP LaserJet supports a powerful set of macro commands. A macro consists of a set of commands recorded in printer memory and identified in all subsequent operations by a macro ID number. Using QuickBASIC, the Macro ID is specified by the sequence

```
ESC$;"&f#Y";
```

where the # is replaced with any number from 0 to 32,767. (Outside a BASIC program, the assignment escape sequence is simply Esc&f#Y). Memory permitting, up to 32 macros can be defined for Series II LaserJets, and up to 32,767 can be defined with the IIP, IID, III, and IIID.

Once the Macro ID number has been specified, a series of macro control commands is used to define, enable, and delete the macro. The Start Macro sequence

```
ESC$;"&f0X";
```

tells the printer to record all commands that follow and associate them with the macro ID number last specified. With only a few exceptions—printer reset, display functions, and macro commands other than call or execute—almost any command can be included. (Note, however, that in the Series II and earlier, no font management commands, such as download, delete, or make font permanent, are allowed.) One macro can call or execute another macro, though only two levels of nesting are permitted.

Sending the End Macro sequence

```
ESC$;"&f1X";
```

signals the end of the macro, which is now defined and ready to be used. The macro can be made permanent (until the printer is turned off) by issuing the command

```
ESC$;"&f10X";
```

This protects the macro against being accidentally deleted by a Reset command.

HP LaserJet macros can be used in three different ways—Execute, Call, and Automatic OverLay—each of which produces a different effect on the current printer environment settings. Since all macro commands operate on the last macro ID specified, you must remember to send the

```
ESC$;"&f#Y";
```

sequence with the appropriate # before using any of the following commands.

The operation of Execute Macro is the easiest to understand. When a QB program sends the

```
ESC$;"&f2X";
```

sequence, the commands associated with the current macro ID number are executed, and any changes the macro makes to the printer environment remain in force after the macro concludes. Thus, for example, you would use the Execute Macro command to change a font and margins for a block of text that would follow. Another macro could then be executed to return the original settings.

HP LaserJet macros can be used in three ways—Execute, Call, and Automatic OverLay—each of which produces a different effect on the printer settings.

Like the Execute Macro command, the Call Macro sequence

```
ESC$;"&f3X";
```

can change the environment (margins, font, and so on), but in this case the changes are discarded and the previous environment is restored when the macro is completed. The only exception is the cursor position, which is not automatically

saved and restored by Macro Call. However, by issuing the Push Cursor command

```
ESC$;"&f0S";
```

at the start of the macro and the Pop Cursor command

```
ESC$;"&f1S";
```

at the end of the macro, the cursor position is maintained. The Push and Pop commands can be placed outside the macro call, but it will save time and memory if you include them in the macro itself. A called macro is usually a self-contained set of commands, such as those for printing a logo.

The Enable Automatic Overlay command

```
ESC$;"&f4X";
```

is similar to the Call Macro command, except that it automatically executes the macro on each page. In the operational sequence, the page is first printed, but before the page is ejected, the Overlaid macro is executed. Automatic Overlay Macros follow the same rules as the Called macro, and are useful for printing headers and footers. To discontinue the operation of an Automatic Overlay macro, the

```
ESC$;"&f5X";
```

command sequence is used.

The program listed in Figure 13 demonstrates the use of macro (and other) commands to create a letterhead for a fictional company. Printing letterheads is one of the most popular uses for macros, and a number of variations have been published in various of the HP *LaserJet Technical Reference* manuals. The cursor movement commands in the program in Figure 13 assume a 14-point font, but you can adjust them as needed. The program also assumes that the font to be used has already been downloaded to the printer (as with the program shown in Figure 12).

DEBUGGING

To err is human, debugging is an important part of working with any language. A simple typo that goes unnoticed can bring down a whole program. Debugging

LETTERHD.BAS — **COMPLETE LISTING**

```
DEFINT A-Z
WIDTH LPRINT 255
ESC$ = CHR$(27)
LPRINT ESC$; "E";
LPRINT ESC$; "&f1y0x0S";      'Macro #1, start macro, push cursor
LPRINT ESC$; "&l0e0L";        'top margin = 0, perf skip off
LPRINT ESC$; "*p75x150Y";     'locate cursor .5" down and in from
                              '  left
LPRINT ESC$; "(54X";          'select font 54 (previously
                              '  downloaded)
LPRINT "VaporWare INC.";      'company name to appear over line
LPRINT ESC$; "*p75x+10Y";     'locate cursor .5" in from left
LPRINT ESC$; "*c2200a30b30g2P"; 'make a line 1/10" tall, 7" long,
                                '  30% shaded
LPRINT ESC$; "*p+75Y";         'move cursor down 1/4"
LPRINT "One Park Avenue, New York, NY  10001" 'address
LPRINT ESC$; "&f1s1X";        'pop cursor and end macro
LPRINT ESC$; "&f4X";          'set as automatic overlay

FOR X = 1 TO 3                'print three pages to show it works
    LPRINT ESC$; "&a30c30R";        'locate cursor center page
    LPRINT "This is page";STR$(X)   'print a message
    LPRINT CHR$(12);                'form feed
NEXT X
```

Figure 13: Automatically printing a letterhead on each sheet is one of the most popular uses for PCL macros. This program generates a letterhead (in 14-point type) for a fictional company.

HP LaserJet escape codes can be tedious, especially when they are incorporated into a large program. As mentioned earlier, the HP LaserJet examines incoming data, and if it finds an escape character, it interprets the following character sequence for further instructions. But what happens if the escape sequence is wrong or outside the range allowed for that command?

An invalid escape sequence gets printed instead of executed, which makes tracking down the error easy.

Fortunately, when an escape sequence is invalid, it gets printed instead of executed, which makes tracking down the error easy. It's harder to find the bug when the sequence is valid, but the data is not. Thus, the sequence

```
ESC$;"&a320C";
```

will be accepted, but unless the HMI is very small, the cursor ends up at the limits of the Logical Page and will appear not to have been executed. Escape sequences that are valid in format but not supported by the printer are simply ignored. The Rotate Text command, for example, is ignored by a Series II printer, since that command is supported only on the LaserJet III.

If you suspect a problem within a section of code, you can use two commands to help find the errant sequence. The Enable Display Function

```
ESC$;"Y";
```

and the Disable Display Function

```
ESC$;"Z";
```

(EscY and EscZ outside BASIC) will cause all escape sequences and control codes to be printed instead of being interpreted and executed. Note that the display functions should be used in conjunction with the PC-8 symbol set and with the end-of-line wrap enabled.

Two other commonly encountered problems show up on the printer display panel: Error 20, MEM OVERFLOW, and Error 21, PRINT OVERRUN. MEM OVERFLOW is caused by a program sending too many macros, too many soft fonts, or too much graphics data. When an Error 20 occurs, the only way out is to clear memory by powering the printer off and back on. The remedy is simply to add more printer memory or to reduce the number of macros, fonts, or graphics.

PRINT OVERRUN is reported when the data sent by a program is too complex

Computers, FCC Class A, Class B, and You — or When is it better to get a B than an A?

You need to know the difference between computers that meet the FCC class B radio frequency emissions standards and those that meet only the Class A standards.

Computers emit radio signals in their operation. Because these signals may cause interference to radio and television reception, the marketing and the use of computers is regulated by the Federal Communications Commission. Under federal rules, computer users are responsible for remedying interference, including interference in neighboring homes.

Computers certified by the FCC as meeting the Class B standard are less likely to cause interference to radio and TV reception than those that have been verified by the manufacturer or importer to the Class A standards. Only Class B certified computers may be advertised, sold, or leased for use in residences. A similar regulatory program applies in Canada.

Buyers seeking computers for use in homes (including offices at home) should shop for computers and peripherals which have been Class B certified. These devices carry a label with an FCC ID number. Both new and used Class A verified devices may be sold only for use in commercial and industrial locations. Signals from computers are more likely to be masked by electrical noise from other equipment in such an environment. These areas are also likely to have fewer radios and TVs. Accordingly, equipment marketed only for use in these locations may meet the less rigorous Class A standard. Class B certified equipment may be marketed for use in residences as well as commercial and industrial locations.

As you shop for a computer for use in your home, look for the FCC classification in the specifications or ask your vendor to recommend only machines that have been certified to the Class B limits. TV viewers and radio listeners in your home and in neighboring homes will be glad you did.

Lab Notes

for the printer to process. Data sent to the printer is stored in an intermediate format. It is then processed concurrently with the actual physical printing you can hear going on. An Error 21 occurs when the printer cannot process the data fast enough to keep up with the physical speed of the page as it moves through the printer. A frequent cause of PRINT OVERRUN when printing graphics is that the program sends commands to print a single point many times during the page run. The order in which printed objects are presented can also cause problems, as when the program sends data for a page and, near the end, has to try to print an object back at the top of the page. If the data is too complex, the portion of the page it tries to print to has already passed by the print drum. By rearranging the way objects are printed, this error can be avoided.

If sufficient extra memory is available, LaserJet III owners can use the Page Protect feature to avoid possible PRINT OVERRUN errors. By selecting the PAGE PROTECT=[page size]*, on t front panel, the HP LaserJet will process the complete page before it actually starts printing. The page size parameter is selected for your current paper size (LTR, Legal, A4 and so on).

Following is a list of telephone numbers for additional information and technical help:

HP FIRST—Fax Information Retrieval Support Technology, (208) 344-4809, lets you receive ap-notes, spec sheets, and more on your fax machine.

HP Personal Peripherals Assist Line—(208) 323-2551, provides general information and tech support.

HP Sales Information—(800) 752-0900, will provide product information and the address of your nearest dealer.

The HP Forum on CompuServe—GO HP, or GO CIS:HP from PC MagNet.

The examples and escape sequences presented here are a bare introduction to the many commands and features of the HP LaserJet Printer Command Language. With creative thinking and an understanding of how to use its language effectively, there is virtually no limit to what can be done with a LaserJet in every office.

Jay Munro is a staff programmer and graphic artist at Crescent Software.